실험 PRJ-1 2주차 기본 테트리스 프로그램 결과보고서

전공: 컴퓨터공학과 학년: 2 학번: 20211599 이름: 주현수

1.

랭킹시스템을 구현하기 위해서 linked list를 선택하였다. 데이터를 크기 순으로 삽입하는 것이나 특정 데이터를 삭제하고 싶을 때, 배열을 사용하면 find ,delete 후 다시 한칸씩 뒤로, 혹은 앞으로 땡겨야 하는 과정이 있는데 이것과는 달리 linked list는 그럴 필요가 없다는 점에서 훨씬 효율적이다. 이는 뒤에 각 함수에서 시간 복잡도를 확인하면서 구체화 하겠다.

Linked list는 name과 score를 갖는 struct로 이루어져 있고 맨 처음 노드를 지칭하는 first가 있다. 매 함수마다 현재 노드를 말하는 curr을 설정했고 필요에 따라 prev라는 curr노드 전 노드를 가르치도록 했다.

전체적으로 rank관련 알고리즘은 다음과 같이 돌아간다.

메뉴 창에서 createRankList()함수를 이용해 rank.txt를 읽어들여 첫줄에 입력된 정보 개수와 각각의 이름과 점수를 읽어서 이를 linkedlist으로 저장하도록 한다. 이때 linkedlist에 저장하는 addrank()함수를 따로 빼줘 코드를 보기 쉽게 정리했다. 메뉴로 들어와서 2번을 누르면 rank()함수 안으로 들어와 그 안에서의 서브 메뉴를 선택하는 항목을 띄운다. 이번 실습에서는 1번을 누를 경우만 했으니 이에 대해서만 설명하도록 하겠다. 1번을 누르면 사용자가 입력한 경우에 따라 linkedlist를 출력하는 과정을 수행한다. 그리고 전체 메뉴에서 1번 을 눌러 play를 하면 맨 마지막에 이름을 받아 새롭게 랭킹을 한다. 이때, newrank()함수를 사용해 새로운 정보를 받고 writeRankFile()을 사용해 새로운 정보를 rank.txt파일에 갱신해주는 과정을 따른다.

1. createRankList

이는 rank.txt파일에서 정보를 읽어들이는 함수이므로 rank.txt의 줄 수만큼의 시간복잡도를 가지고 있다.

1. addrank()

이는 크기 순으로 내림차순 정리를 해야하므로 최악의 경우로 현재 linkedlist에 저장된 노드의 개수만큼 프로그램이 돌 수 있다. 이를 linkedlist가 아닌 배열로 썼다면 최악의 경우 (linkedlist에 저장된 노드의 개수 )\*2 만큼의 시간복잡도를 가질 수 있다. 하나 추가했다면 뒤로 밀어줘야하기 때문이다.

1. rank() – 1번 입력했을 시 상황

출력해야하는 상황이니 rank.txt의 줄 수만큼 시간복잡도가 걸린다.

1. writeRankFile()

rank.txt파일에 저장해야하는 거니 linkedlist의 노드개수 + 1(size) 만큼의 시간복잡도가 걸린다.

1. newRank()

addRank()와 writeRankFile() 의 시간복잡도를 더한만큼의 시간복잡도가 걸린다.

2.

이번 주차 실습+ 과제를 하면서 linkedlist의 자료구조를 활용할 때는, 특히 프로그램이 클수록 core dumped오류가 정말많이 나는 것을 알았다. 자료의 추가와 삭제를 위해서는 편리한 자료구조이지만 적절할 때 free를 해줘야하고, 삭제와 삽입 시, 노드가 하나도 없거나 하나만 있는 첫 경우와 그렇지 않은 경우를 나눠야 하며, curr와 prev의 위치, 코드의 순서에 따라서 프로그램이 크게 바뀌는 것을 습득했다. 또한 프로그램을 짜기 위해서 어느 자료구조가 더 효율적인지 파악을 하는 과정이 필요함을 알게 되었다. 자료구조의 형태를 아는 것이 얼마나 중요한지를 깨닫게 되었다.

<과제>

1.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

없는 정보 들어갔을 때

정보가 잘 들어갔을 경우

시간복잡도는 최악의 경우 linkedlist에 저장돼 있는 노드의 개수만큼이다.

2.

